Please type a plus sign (+) inside this box → ☐

HDP/SB/21 based on PTO/SB/21 (08-00)

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Application Number	**09/495,036**
Filing Date	January 31, 2000
Inventor(s)	Nancy D. GRIFFETH et al.
Group Art Unit	2665
Examiner Name	Justin Philpott
Attorney Docket Number	29250-000920/US

ENCLOSURES *(check all that apply)*

- ☒ Fee Transmittal Form
  - ☒ Fee Attached
- ☐ Amendment
  - ☐ After Final
  - ☐ Affidavits/declaration(s)
- ☐ Extension of Time Request
- ☐ Express Abandonment Request
- ☐ Information Disclosure Statement
- ☐ Certified Copy of Priority Document(s)
- ☐ Response to Missing Parts/ Incomplete Application
- ☐ Response to Missing Parts under 37 CFR 1.52 or 1.53
- ☐ Assignment Papers *(for an Application)*
- ☐ Letter to the Official Draftsperson and _____ Sheets of Formal Drawing(s)
- ☐ Licensing-related Papers
- ☐ Petition
- ☐ Petition to Convert to a Provisional Application
- ☐ Power of Attorney, Revocation Change of Correspondence Address
- ☐ Terminal Disclaimer
- ☐ Request for Refund
- ☐ CD, Number of CD(s) _____
- ☐ After Allowance Communication to Group
- ☐ LETTER SUBMITTING APPEAL BRIEF AND APPEAL BRIEF (w/clean version of pending claims)
- ☒ Appeal Communication to Group *(Notice of Appeal, **Brief**, Reply Brief)*
- ☐ Proprietary Information
- ☐ Status Letter
- ☐ Other Enclosure(s) *(please identify below)*:

Remarks

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm *or* Individual name	Harness, Dickey & Pierce, P.L.C.	Attorney Name John E. Curtin	Reg. No. 37,602
Signature			
Date	April 29, 2005		

PTO/SB/17 (12-04)
Approved for use through 07/31/2006. OMB 0651-0032
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

FEE TRANSMITTAL for FY 2005

Effective 10/01/2004. Patent fees are subject to annual revision.

☒ Applicant claims small entity status. See 37 CFR 1.27

TOTAL AMOUNT OF PAYMENT ($) 500

Complete if Known	
Application Number	09/495,036
Filing Date	January 31, 2000
First Named Inventor	Ruibing HAO
Examiner Name	Justin M. Philpott
Art Unit	2665
Attorney Docket No.	29250-000920/US

METHOD OF PAYMENT *(check all that apply)*

☐ Check ☐ Credit card ☐ Money Order ☐ Other ☐ None

☒ Deposit Account:

Deposit Account Number: 08-0750

Deposit Account Name: Harness, Dickey & Pierce, PLC

The Director is authorized to: *(check all that apply)*
☐ Charge fee(s) indicated below ☐ Credit any overpayments
☐ Charge any additional fee(s) during the pendency of this application
☐ Charge fee(s) indicated below, **except for the filing fee** to the above-identified deposit account.

FEE CALCULATION

1. BASIC FILING FEE

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
1011	300	2011	150	Utility filing fee	
1012	200	2012	100	Design filing fee	
1013	200	2013	100	Plant filing fee	
1014	300	2014	150	Reissue filing fee	
1005	200	2005	100	Provisional filing fee	

SUBTOTAL (1) ($) 0

2. EXTRA CLAIM FEES FOR UTILITY AND REISSUE

			Extra Claims		Fee from below		Fee Paid
Total Claims		-20 ** =	0	X		=	0
Independent Claims		-3 ** =	0	X		=	0
Multiple Dependent						=	0

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description
1202	50	2202	25	Claims in excess of 20
1201	200	2201	100	Independent claims in excess of 3
1203	360	2203	180	Multiple dependent claim, if not paid
1204	200	2204	100	** Reissue independent claims over original patent
1205	50	2205	25	** Reissue claims in excess of 20 and over original patent

SUBTOTAL (2) ($) 0

***or number previously paid, if greater; For Reissues, see above*

FEE CALCULATION (continued)

3. ADDITIONAL FEES

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
1051	130	2051	65	Surcharge - late filing fee or oath	
1052	50	2052	25	Surcharge - late provisional filing fee or cover sheet.	
1053	130	1053	130	Non-English specification	
1812	2,520	1812	2,520	For filing a request for reexamination	
1804	920*	1804	920*	Requesting publication of SIR prior to Examiner action	
1805	1,840*	1805	1,840*	Requesting publication of SIR after Examiner action	
1251	120	2251	60	Extension for reply within first month	
1252	450	2252	225	Extension for reply within second month	
1253	1020	2253	510	Extension for reply within third month	
1254	1,590	2254	795	Extension for reply within fourth month	
1255	2,160	2255	1080	Extension for reply within fifth month	
1401	500	2401	250	Notice of Appeal	
1402	500	2402	250	Filing a brief in support of an appeal	500
1403	1000	2403	500	Request for oral hearing	
1452	500	2452	250	Petition to revive – unavoidable	
1453	1500	2453	750	Petition to revive – unintentional	
1501	1400	2501	700	Utility issue fee (or reissue)	
1502	800	2502	400	Design issue fee	
1460	130	1460	130	Petitions to the Commissioner	
1807	50	1807	50	Processing fee under 37 CFR 1.17 (q)	
1806	180	1806	180	Submission of Information Disclosure Stmt	
8021	40	8021	40	Recording each patent assignment per property (times number of properties)	
1809	790	2809	395	Filing a submission after final rejection (37 CFR § 1.129(a))	
1810	790	2810	395	For each additional invention to be examined (37 CFR § 1.129(b))	
1801	790	2801	395	Request for Continued Examination (RCE)	
				Other fee (specify) ____	

***Reduced by Basic Filing Fee Paid** **SUBTOTAL (3)** ($)500

4. SEARCH/EXAMINATION FEES

1111	500	2111	250	Utility Search Fee
1112	100	2112	50	Design Search Fee
1113	300	2113	150	Plant Search Fee
1114	500	2114	250	Reissue Search Fee
1311	200	2311	100	Utility Examination Fee
1312	130	2312	65	Design Examination Fee
1313	160	2313	80	Plant Examination Fee
1314	600	2314	300	Reissue Examination Fee

SUBTOTAL (4) ($)0

SUBMITTED BY

Complete *(if applicable)*

Name (Print/Type)	John E. Curtin	*Registration No. (Attorney/Agent)*	37,602	*Telephone*	(703) 668-8000
Signature				*Date*	April 29, 2005

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Appeal No. _____

Appellants: Ruibing HAO et al.

Application No.: 09/495,036

Group No.: 2665

Filed: January 31, 2000

Examiner: Justin M. Philpott

For: GENERATION OF TEST SUITES FOR INOPERABILITY OF REACTIVE COMMUNICATION SYSTEMS

Attorney Docket No.: 29250-000920/US

BRIEF ON APPEAL ON BEHALF OF APPELLANT

Customer Service Window
Randolph Building
401 Dulany Street
Alexandria, VA 22314
Mail Stop Appeal Brief - Patents

April 29, 2005

TABLE OF CONTENTS

BRIEF ON BEHALF OF APPELLANT

In support of the Notice of Appeal filed March 29, 2005, appealing the Examiner's final rejection mailed December 29, 2004 of each of pending claims 3-6 and 8-10 of the present application which appear in the attached Appendix A, Appellant hereby provides the following remarks.

I. REAL PARTY IN INTEREST

The present application is assigned to Lucent Technologies Inc., by an Assignment recorded on May 10, 2000, Reel 010805, Frame 0134.

II. RELATED APPEALS AND INTERFERENCES

The Appellant does not know of any appeals or interferences which would directly affect or which would be directly affected by, or have a bearing on, the Board's decision in this Appeal.

III. STATUS OF THE CLAIMS

The claims reproduced in the attached Appendix A are the claims on Appeal. Each of these claims is currently pending in the application.

IV. STATUS OF ANY AMENDMENTS FILED SUBSEQUENT TO THE FINAL REJECTION

A Request for Reconsideration dated March 3, 2005 was filed with the U.S. Patent and Trademark Office in response to the Final Rejection and was considered and entered according to the Advisory Action. The Advisory Action indicates that claims 3-6 and 8-10 are rejected.

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

It is believed that Internet protocol (IP) will play a large, if not dominant, role in future public telephony networks. Such networks are expected to handle not only voice, but video and other information and data as well. It is therefore important that existing data-centric and public switched telephone networks (PSTNs) be capable of operation when connected with one another through so-called "gateways". The ability of such networks or systems to operate properly when inter-connected is typically referred to as system "interoperability" (see specification, pp. 1-2).

With respect to providing voice calls over an IP network ("Voice over IP" or "VoIP"), the first VoIP systems provided voice communications over a local area network (LAN) for end-users seated at personal computers. This proved unsatisfactory for several reasons. For example, voice quality was poor due to a "best-efforts" standard for data packet delivery in a LAN environment. Also, there was no way to direct a voice call to a user connected to a PSTN, thus limiting the number of possible calls (excerpts from specification, p. 2).

Because some providers believed IP telephony might circumvent long distance user rates due to the absence of tariffs on local Internet access, local user access gateways to PSTNs were created in each locality to be served, with a private Internet providing a long-distance backbone for voice calls. LEC's became interested in a similar strategy, but instead of using a gateway to connect local lines to an IP network, gateways were used to connect their

central offices to an IP network, thus replacing the circuit-switched network backbone with an IP network. Finally, corporations maintaining large data networks became interested in combining their voice and data networks using Voice over IP.

In view of the above, the existing diversity of VoIP implementations results in diverse communications systems that may not be able to interoperate when connected together. In order for these existing communication systems to remain effective, they must be able to interwork with the PSTN and with other VoIP products.

The Standards

Three important Internet telephone standards now exist. Initially, Standard H.323 for Internet Telephony reflected the LAN bias of earlier Internet telephony systems. Subsequent standards address the use of packet voice over wide-area networks and interwork with the PSTN.

Standard H.323 version 2 is a current dominant standard for commercial products. A second standard, Session Initiation Protocol (SIP), is a proposed standard of the Internet Engineering Task Force. A third standard, Media Gateway Control Protocol (MGCP), has also been proposed to provide interworking between Internet telephony and the PSTN.

The H.323 Standard is both an architecture and a standard for Voice over IP. It is an umbrella standard, specifying a collection of standards to be used by the components of an architecture. It divides required functionality

among gatekeepers, gateways, multipoint control units (MCUs), and endpoints (excerpts from specification, pp. 2-4).

Specifically, so-called endpoints are the initial creators and ultimate recipients of information streams. They need not comply with any part of the H.323 protocol, but if they do not, they must communicate with a network through a gateway that does comply with the protocol. An endpoint that complies with H.323 15 is called an H.323 terminal.

An H.323 gateway provides communications between an H.323 network and a non-H.323 device or network. It may also provide communications with a second H.323 network.

An H.323 gatekeeper handles address translations, for example, between telephone numbers and IP addresses, and controls access to the network. A gatekeeper controls admission to a zone. Gateways, MCU's, and H.323 terminals must register with a gatekeeper, if one is present in their zone, to obtain permission to join a call. Optionally, call signaling may also go through the gatekeeper. There is no gatekeeper-to-gatekeeper standard, so that communications between multiple zones must be managed in an ad hoc fashion.

A multipoint control unit provides conferencing capabilities in an H.323 network. It includes a multipoint controller, and (optionally), one or more multipoint processors. The multipoint controller handles signaling that determines who participates in a conference, and what information streams

they send and receive. The multipoint processors provide centralized processing of information streams (audio, video, or data) from various parties to a conference. The processing can be mixing, switching, or anything supporting a specific conference connection.

An H.323 terminal or gateway uses a Registration, Admission, and Status (RAS) channel defined in the standard to register with a gatekeeper, to locate other gateways and terminals, and to request permission to start or join a call. It may also route the call signaling through the gatekeeper. The call signaling protocol is a Q.931 subset, also defined in the H.225.0 Standard.

The actual media connections in a call and their types, are negotiated directly between gateways and/or endpoints using the H.245 Standard. Once the endpoints agree, transport connections are also set up using the H.245 standard. Transport connections can use any of a number of standards. For audio, the G.711 Standard for uncompressed voice is required, but not always implemented. Other, optional audio Standards are G.723.1 and G.729. For video, the H.261 Standard is required, while the H.263 Standard is optional. For data, the T.120 Standard is required (excerpts from specification, pp. 4-6).

Because of the many choices available when building an H.323-compliant system, and because the requirements for communications between zones are not specified, it is quite likely that H.323-compliant

systems manufactured by different vendors will not interoperate. Further, the number of configurations to be tested to determine system interoperability, becomes relatively large.

The Session Initiation Protocol (SIP) identifies the following functionality which is needed to set up multimedia conferences:

User Location: Determination of the end system to be used for communication. A location server provides user location information, in response to a request from a SIP server or proxy. Users may also register their locations with a SIP server, using a REGISTER method.

User Availability: Determination of the willingness of a called party to engage in communications. An end-user wishing to set up a call with another end-user, or to add it to a conference, invokes an INVITE method on the user's SIP server. Possible responses include "Success" and "Redirection".

User Capabilities: To determine what media and media parameters will be used for a call, the calling system invokes an OPTION method on the SIP server of the called party. This is an inquiry as to what are the capabilities of the called party.

Call Setup: "Ringing", establishment of call parameters at both called and calling party (excerpts from specification, pp. 6-7).

The inoperability of SIP implementations is less problematic than that of H.323 implementations. Interoperation of SIP user agents and servers with H.323 networks remains problematic, however. A type 7R/E Programmable

Feature Server, available from Lucent Technologies Inc., provides protocol interworking through device servers that interface to specific protocols.

The Media Gateway Control Protocol (MGCP) is a combination of two earlier standards, IP Device Control (IPDC) and SGCP (Simple Gateway Control Protocol SGCP). This protocol provides for media gateways that convert signals from TDM to packet, and back, and for call agents that control the behavior of media gateways. This is a master/slave arrangement, with the call agent as the master and the gateway as the slave. Synchronization between various call agents involved in a call must be handled by other means.

MGCP is thus used as an internal protocol within a distributed system that appears as a single VoIP gateway to an external H.323 network. A call agent may signal to other call agents using the RAS and H.225.0 protocols. This provides a means to interwork H.323 networks with IP networks.

Accordingly, there is currently a need to verify that when two communication systems are connected for interworking, such as to provide, e.g., VoIP service, the connected system(s) behavior will be as expected from interactions with one or more components of the integrated system under test. Specifically, there is a need for a method in the form of, e.g., portable software, that will generate test sequences which, if successfully implemented, will assure both branch and full path coverages for the connected systems (excerpts from specification, pp. 7-8).

A model 10 of a general configuration that allows communication systems to interoperate, such as Voice over IP systems, is shown in FIG. 1 (Appendix B). FIG. 1 shows two end-users 12, 14 that want to communicate. Each user can access the systems through a corresponding gateway 16, 18. The two gateways 16, 18 communicate with each other using a defined protocol to decide whether and how to start the communication between the end-users 12, 14. The protocol takes the gateways 16, 18 through various states as they negotiate concerning the desired communication. The end users 12, 14 may wish to communicate by voice, or to exchange other information in either analog or digital form (excerpt from specification, p. 12).

It is important to note that interoperability errors, if any, will be introduced only when the gateways 16, 18 actually communicate with one another about a call. Local activities involved in the protocol, e.g., obtaining information from either end-user 12 or 14 can be ignored. Interactions or "transitions" that can be ignored for interoperability purposes are referred to as "white", and all other transitions are referred to as "black". A white transition is purely local, that is, it reads the state of only one gateway 16 or 18, and writes the same state. A black transition involves both gateways 16 and 18 because the transition reads the states of both of the gateways, or reads the state of one and writes the state of the other.

For example, an "off-hook" transition from an idle state of a user's telephone to a dialing state of the same user's telephone is white, because the transition involves only an originating gateway and its associated end user. A "dial" transition from "dialing" of a calling user's telephone to "ringing" of a called user's telephone is black, because the states change in both gateways 16 and 18. Similarly, an off-hook transition that connects the call changes states in both gateways 16, 18 from ringing to connected. Accordingly, test cases that correspond only to sequences involving black edges are generated, i.e., those sequences that occur while a call is under negotiation between the two gateways 16 and 18.

Expected system behavior is modeled in accordance with a finite state machine (FSM) 50, typical vertices (nodes) and edges of which are shown in FIGs. 2, 2A-2D (Appendices C-G). The FSM 50 of FIGs. 2, 2A-2D have 21 states (nodes) and a total of 68 transitions between the states, as defined in FIGS. 3-9 (Appendices H-N). A transition from a first state to a second state is identified by locating the two ordered states on the first line of one of the 68 transitions in FIGS. 3-9. Ideally, all possible execution sequences or "scenarios" should be covered. Because the transition diagram of the FSM 50 is a directed graph covering all possible execution sequences requires that all branches and all possible paths be tested. Criteria for ruling out "redundant" scenarios are given further below, however (excerpts from specification, pp. 12-14).

Generally, the transition diagram will contain cycles, and, therefore, will have an infinite number of distinct paths. Therefore, the test generation process includes the following three steps:

Test Generation Process

Step 1: Generate all possible acyclic paths, i.e., paths without repeated vertices. See FIGS. 10 to 13 (Appendices O-R).

Step 2: Generate all possible simple cycles, i.e., cycles that do not contain any smaller cycles. See FIG. 14 (Appendix S); and

Step 3: "Combine" the paths (from Step 1) and cycles (from Step 2) to generate a final set of paths. See FIG. 15 (Appendix T).

The number of paths and cycles generated in the first two steps is finite. Various criteria for removing redundant acyclic paths and redundant simple cycles are given later below.

With respect to Steps 1 and 2, all strongly connected components (SCC) of the transition graph are first found. This has two advantages; (a) because it is known that any cycle is completely contained inside a SCC, Step 2 can be performed by looking at each SCC in turn and finding all simple cycles within the SCC, and (b) each SCC can be "shrunk" into a node to obtain a Directed Acyclic Graph (DAG), i.e., a graph without any cycles. This translates into a two-phase process for Step 1. First, generate all acyclic paths on the resulting DAG, and then replace each SCC on any given path with a set of acyclic paths within the SCC.

The goal of the "combine" process in Step 3 is to generate a finite number of paths that cover all scenarios of interest. This is done first by including all the acyclic paths. Then additional cyclic paths are generated as follows.

For each acyclic path P, find all the cycles that share a node with P. Let these cycles be $C_1, C_2, \ldots C_k$, and let v_i, $1 \le i \le k$, be a node common to C_i and P. Then generate a new (cyclic) path by replacing node v_i in P with cycle C_i (excerpts from specification, pp. 14-15).

Next-Transition-Tree

A simple data structure, next-transition-tree, which is convenient for Steps 1 and 2, is now described. See FIG. 12 (Appendix Q). The data structure can be defined for any graph but, in the present application, the graph is always a SCC.

For any node v, next-transition-tree(v) stores all acyclic paths from v to other vertices in its SCC. The tree has v as its root, and its height is equal to the number of nodes in the SCC containing v. Children of v are all the nodes in its SCC that have a direct edge from v. In general, the children of any node p are all the nodes in its SCC that have a direct edge from node p. Note that a node may appear multiple times in this tree. That is, each node has a label where the labels are not necessarily unique. The root node has label v. A node labeled μ has as many children as the outdegree of u (number of edges

leaving *u*) in its SCC, and these children are labeled with the corresponding nodes in the SCC.

For ease of illustration, assume that a separate next-transition-tree is maintained for each node in the graph. The actual implementation may have many shared pieces among next-transition-trees belonging to the same SCC.

Generating All Simple Cycles

All simple cycles containing any node *v* are generated as follows: Consider all paths in next-transition-tree(*v*) that contain another (than root) instance of node *v*. In any such path, the path segment from the root to the first (closed from root) occurrence of v corresponds to a cycle containing *v*. This path segment will correspond to a "simple" cycle if it does not contain any repeated nodes. Therefore, the process includes finding all such path segments, and all simple cycles containing *v* are generated. As mentioned, this is a simplified description of an actual implementation, which should not maintain any paths with repeated nodes in the next-transition-tree (excerpts from specification, pp. 15-17).

One way to generate all simple cycles in a SCC is to consider its vertices in some order: $v_1, v_2, \ldots, v_k$. Generate all simple cycles containing the node v_1. Then generate all simple cycles containing the node v_2 that don't contain node v_1. This can be accomplished by modifying the above procedure so that path segments containing the node v_1 are ignored. All simple cycles

containing the node v_3 that don't contain nodes v_1 or v_2, are then generated, and so on.

Generating All Acyclic Paths

As mentioned earlier, all the strongly connected components (SCC) of the transition graph must be found. A "component graph" where each node represents a SCC, and an edge from node u to node v represents an edge (in the original transition graph) from a node belonging to u's SCC to a node belonging to v's SCC, is then constructed. This component graph is a Directed Acyclic Graph (DAG), i.e., a graph without any cycles. See A.V. Aho, et al., The Design and Analysis of Computer Algorithms (Addison-Wesley 1974), all relevant portions of which are incorporated by reference. All acyclic paths in the DAG are first generated, and then each SCC on any given path is replaced with a set of acyclic paths within the SCC (excerpts from specification, p. 18).

Redundancy

There are a finite number of distinct paths representing potential tests in a DAG. Not all of them need to be tested, however. Rather, a minimal number of tests, which collectively satisfy a coverage criterion for the connected systems and which do not contain any redundant tests, are desirably generated.

The tests must start from a "Start" state, so only those paths beginning from a source node which has no incoming transitions are considered first. The following redundancy criteria may then be applied:

(R1) Proper prefixes are redundant and may be ignored.

Redundancy criterion R1 implies that only those paths terminating at a sink node (e.g., a node without an outgoing transition) be considered. Any path not terminating at a sink node is a proper prefix of a path that has been extended to a sink node. Accordingly, only those paths that start from the source node and terminate at one of the sink nodes need to be generated.

The following procedure generates all source-to-sink paths in a DAG.

PATHS-IN-DAG

Input. **a DAG *G* with one sources and multiple sinks.**
Output. **all source-sink paths in *G*.**

```
1    topologically sort nodes in G: s = v0, v1,...,vn-1;
2    for i = n-1,...,0
3        if vi is a sink node then
4            p(vi) = {Λ}; /* a singleton set of empty path */
5        else
6            let outgoing edges from vi be: w1,..., wr;
7            p(vi) = ∪_{j=1}^{r} (v_i, w_j) p(w_j);
8    return p(v0)
```

The above procedure is bottom-up, starting from one of the sink nodes $t = v_{n-1}$. When processing a node v_i, we examine all of its outgoing edges (v_i, w_j)

where the paths from w_j to sink nodes have been computed. At Line 7 above, we concatenate edge (v_i, w_j) to each path computed at w_j and collect it at node v_i. After processing the source node $s = v_0$, all of the paths which are irredundant and have complete coverage are obtained (excerpts from specification, pp. 18-20).

A topological sort takes time proportional to the number of edges. We charge to the examination and concatenation of each edge, which is processed only once at Line 7, and the total cost is proportional to the total paths lengths.

Proposition 1. The procedure PATHS-IN-DAG constructs all source-sink paths. Its time and space requirement is linear in output, i.e., the total lengths of all the constructed tests.

Proposition 2. For machines with a reset to source s, the constructed test sequences are a checking sequence.

Generating All Acyclic Paths within a SCC

The generation of paths (i.e., tests) for DAGs where nodes can be SCC's has been discussed. For each edge connecting two SCC's on a path, the edge is replaced with an edge in the original graph. In general, it will be possible to replace an edge between SCC's with one of several possible edges in the original graph, and a separate path for each choice of replacement edge can be obtained (excerpts from specification, pp. 18-20.

Each SCC node is also replaced with a set of acyclic paths within the SCC. Assume the incoming and the outgoing edges to the SCC node are on nodes u and v of the SCC. The SCC node needs to be replaced with all possible acyclic paths from u to v in the SCC. These paths can be generated from next-transition-tree (u).

Replacing any SCC with all possible acyclic path provides exhaustive coverage, but may generate a relatively large number of tests. In certain applications, it may not be necessary to cover the SCC's as thoroughly. Two options used in practice are as follows:

Chinese Postman Tour. For an SCC, each edge (transition) is tested at least once and the test sequence length is minimized. Such a path is called a Chinese Postman Tour. See A. V. Aho, et al., An Optimization Technique for Protocol Conformance Test Generation Based on UIO Sequences and Rural Chinese Postman Tours, 39 IEEE Trans. on Communication, No. 11 (Nov. 1991) at 1604-15, which is incorporated by reference.

Checking Sequence. A more thorough coverage is provided by a checking sequence that guarantees the structural isomorphism of the implementation and the specification machines. The length of such a test sequence will be longer than that of a Chinese Postman Tour.

The following procedure summarizes the steps in generating all possible tests:

TEST-GENERATION

Input. An EFSM with a designated source state
Output. Minimal set of tests providing full coverage
1 Construct an equivalent FSM and its transition diagram G
2 Find all SCC's of G.
3 For each SCC
4 for each node in the SCC
5 construct next-generation-tree
6 Shrink each SCC of G to obtain a DAG G'
7 Apply procedure PATHS-IN-DAG on G' to obtain all acyclic source-sink paths in G'
8 Replace each edge and node to obtain the set P of acyclic paths
9 Obtain the set C of all simple cycles
10 Combine the sets P and C to obtain the final set of tests

Additional Redundancy Criteria

The TEST-GENERATION procedure, above, provides exhaustive coverage. The number of tests generated may be enormous even for moderate sized systems, however (excerpts from specification, pp. 20-23).

As stated earlier, system interoperability errors may occur only when the system gateways actually "talk" to one another concerning a given call. Transitions are therefore labeled as either "white" to connote local activity, or as "black" if they involve both gateways. Thus, for interoperability test generation, white transitions need not be covered, and the following additional redundancy criteria become appropriate:

(R2) Remove all-white test sequences.
(R3) Let (u,v) be the last black edge in the test sequence.
Then replace the path from the source node to node u with the shortest path between the source node and node *u*.
(R4) Let (u,v) be the first black edge in the test sequence.

> Then replace the path from node v to the sink node with the shortest path between node v and the sink node.
> (R5) If there is a sequence of white edges in which each party separately reaches an idle state, terminate the sequence at the last black edge and then use a shortest path to the state "IdleA, IdleB" at the end of the sequence.

Criteria R2, R3 and R4, above, reflect that white transitions are not relevant for interoperability testing. Their only use is to connect relevant black transitions. For example, R3 states that the sequence of transitions before the first black transition is not relevant, so that part may be replaced with the shortest all-white sequence. Criterion R4 is a dual of R3. The last criterion R5 reflects that if both parties reach an "idle" state through a sequence of non-relevant transitions, then nothing concerning system interoperability will happen in the rest of the sequence (excerpts from specification, pp. 23-24).

The following COMPLETE COVERAGE procedure generates all acyclic paths satisfying these criteria:

COMPLETE COVERAGE

Input. **transition graph G of a PSM.**
Output. **complete set of paths according to redundancy criteria *R1-R5*.**

```
1   for each node v that has an outgoing black edge
2      if there is an all-white path from the source node to v then
3        add a super-edge from the source node to v;
4   delete all outgoing edges (not super-edges) from the source node;
5   for each node v that has an incoming black edge
6      if there is an all-white path from v to a sink node then
7        add a super-edge from v to this sink node;
8   delete all incoming edges (not super-edges) to the sink node;
9   generate all acyclic paths in the resulting graph by procedure
    PATHS-IN-DAG with the modification that no super-edge should
    follow or precede a white edge;
10  replace each super-edge with the shortest all white path in the
    original graph;
11  process each path according to redundancy criterion R5;
```

Generating Test Sequences according to Rl-R5

The procedure above starts with a graph G and generates another graph G' such that the set of source-sink paths in G' is the same as the set of source-sink paths in G, while satisfying criteria R2-R4. Lines 1-4 of the procedure above considers all possible black transitions that may be the first in any sequence, and add a marker 'super-edge' for the shortest all-white path from the source node. Line 9 ensures that a black transition follows a super-edge. The marker 'super-edge' is replaced on line 10 by the shortest all-white path. This handles redundancy criterion R3. Lines 5-8 perform an analogous function for sink nodes and the last black transitions in the sequence, reflecting redundancy criterion R4. Criterion R2 is satisfied because line 9 ensures there is at least one black transition.

Instead of generating all paths and then processing them according to criterion R5 per line 11, an actual implementation may ensure that the incremental construction of all paths in procedure PATHS-IN-DAG is such that a path in violation of R5 is never generated.

Proposition 3. The procedure COMPLETE-COVERAGE constructs all acyclic paths according to criteria Rl-R5 (excerpts from specification, pp. 25-26).

Generating a Smaller Set of Test Sequences

On most practical systems, we expect the above COMPLETE-COVERAGE procedure to generate a considerably smaller set of test sequences than the TEST-GENERATION procedure. But even such a smaller set may be too large for manual testing, however. Accordingly, the following describes a set of more restrictive criteria for testing, concentrating only on black transitions.

> (R6) Generate acyclic paths having only black edges, except that the prefix from the source node, and the suffix to the sink node, are allowed to contain white edges.
> (R7) Generate simple cycles having only black edges.
> The following ADEQUATE-COVERAGE procedure generates a test-set according to R6 and R7.

```
Input. transition graph G of a FSM.
Output. test generation according to R6 and R7.
1   full coverage.
2     for each node v that has an outgoing black edge
3       if there is a path from the source node to v then
4         add a super-edge from the source node to v;
5     for each node v that has an incoming black edge
6       if there is a path from v to a sink node then
7         add a super-edge from v to this sink node;
8   delete all white edges from the graph;
    /* the only remaining edges are super-edges or
    black edges */
9   generate all acyclic paths in the resulting
    graph;
10  replace each super-edge with the shortest path in the original
    graph;
```

The ADEQUATE COVERAGE procedure starts with a graph G, and transforms it into a graph G' such that the set of source sink paths in G' is the same as the set of source-sink paths in G with criterion R6. The intuition of the procedure is to delete all white edges except those needed to reach black transitions from the source node, or from the black transitions to the sink nodes. Line + -maintains a marker 'super-edge' for each source node to black transition path. On line 10, this marker is replaced with the shortest path. Lines 4-6 perform an analogous function for sink nodes. Criteria for adequate coverage are represented in FIG. 16 (Appendix U).

Proposition 4. The ADEQUATE-COVERAGE procedure constructs all acyclic paths according to criterion R6.

If an automated test capability is available, then a COMPETE-COVERAGE procedure is more desirable. If tests must be executed by hand, however, then the ADEQUATE-COVERAGE procedure will likely produce a manageable set of test sequences. Alternatively, one may always start with the ADEQUATE-COVERAGE procedure because it does apply to the most critical interoperability behavior. If the systems pass those tests generated by the ADEQUATE-COVERAGE procedure, one may obtain the broader coverage provided by the COMPLETE-COVERAGE procedure (excerpts from specification, pp. 26-28).

EXAMPLE:

A portable software tool referred to as an Interoperability Testing Intelligent System (IT-IS) for automated interoperability test generation, was written in ANSI C and Tcl/Tk. The program includes a graphical user interface (GUI) for user input, and for displaying generated test sequences. The workflow of IT-IS is shown in FIG. 17 (Appendix V).

The input to IT-IS is an extended FSM (EFSM) description of the composed system behavior. IT-IS first performs reachableness analysis to convert the EFSM into a FSM, e.g., FSM 50 in FIGs. 2, 2A-2D (Appendices C-G), and then uses different procedures on the FSM to generate the test sequences.

IT-IS was used to generate interoperability test cases for end-user VoIP testing. As shown in FIGS. 3-9 (Appendices H-N), the FSM 50 has 21 states and 68 transitions.

Among the transitions, 24 are colored black, others are white. The shrunk DAG generated by IT-IS from the FSM 50 contains 3 SCC nodes, and only one of them has more than one state. The number of test sequences generated by applying the procedures described above, is shown in the following table.

Procedure	Loop-Free Paths	Loops	Final Tests
TEST-GENERATION	950	424	1752
COMPLETE-COVERAGE	508	424	908
ADEQUATE-COVERAGE	16	4	22

The final tests for either of the COMPLETE-COVERAGE or the ADEQUATE-COVERAGE procedures involve only black transitions.

Interoperability testing is indispensable for integration of reactive communication systems. The procedures disclosed herein are distinguishable over conventional conformance testing techniques because the procedures disclosed herein focus on system interfaces, and are not directed solely to individual system implementations or specifications (excerpts from specification, pp. 28-29).

Appellants respectfully note that the above summary of the invention, including any indication of reference numerals, drawings, figures, paragraphs, page numbers, etc. (collectively referred to as "descriptions" of the application) have been provided solely to comply with the U.S. Patent and Trademark Office's rules concerning the appeal of the claims of the present application. As such, the descriptions above are merely exemplary and should not be construed to limit the claims of the present application in any way whatsoever.

VI. ISSUES TO BE REVIEWED ON APPEAL

(i.) Whether or not claims 3-6 and 8-10 are anticipated by U.S. Patent No. 6,466,548 to Fitzgerald ("Fitzgerald").

VII. ARGUMENTS

Claims 3-6 and 8-10 were rejected under 35 U.S.C. §102(e) as being anticipated by Fitzgerald. Appellants respectfully disagree for at least the following reasons.

Each of the claims of the present invention requires, among other things:

(a) a method of generating test sequences for *evaluating the interoperability of communications systems* ...(italics added);

(b) determining transitions ... required to implement [a] desired mode of operation, wherein each transition pertains to a first operation of [a] first VoIP gateway and a corresponding second operation of [a] second VoIP gateway...; and

(c) the generation of "acyclic" paths from a transition diagram.

Initially, Appellants note that in the Advisory Action the Examiner primarily relies on the general presumption that the preamble of a claim is not "accorded any patentable weight" to mostly ignore the phrase *evaluating the interoperability of communication systems*. That may be permissible until, as in the instant application, the Applicants/Appellants have relied on such terms for patentability purposes. See for example, *Bristol-Meyers Squibb Co. v. Ben Venue Labs, Inc.* 246 F.3d 1368, 1375, 58 USPQ 2d 1508, 1513 (Fed. Cir. 2001), *Bell Communications Research, Inc. v. Vitalink Communications Corp.* 55 F.3d 615, 620, 34 USPQ2d 1816,1820 (Fed. Cir. 1995) and *Pitney-Bowes Inc. v. Hewlett-Packard Co.* 182 F.3d 1298, 1306 (Fed. Cir. 1999).

The Examiner's failure to give the proper patentable weight to the phrase *evaluating the interoperability of communication systems* is impermissible. See for example, *In re Wilson*, 165 USPQ 494,496 (CCPA 1970) ("All words in a claim must be considered in judging patentability of that claim against the prior art"). Therefore, Appellants respectfully request that the Board reverse the rejections of claims 3-6 and 8-10.

Continuing, Appellants respectfully submit that Fitzgerald does not disclose or suggest features (a) through (c), among other features, set forth above.

Instead, Fitzgerald discloses a technique for isolating a sub-link responsible for causing delays. There is no evaluation of the inter-operability of communications systems disclosed or suggested in Fitzgerald. Rather, in order

to isolate a sub-link, Fitzgerald requires the installation of loopback interfaces at one or more routers. Such a loopback arrangement prevents Fitzgerald from testing the interoperability of its systems.

It also appears that the Examiner is attempting to equate a network topology with the claimed acyclic paths. Appellants respectfully disagree. In accordance with the specification, acyclic paths are defined as those paths that do not have repeated vertices (see Figures 10-13; page 14, lines 15 and 16). One of ordinary skill in the art would not equate the phrase acyclic paths as used in the claims with the general topology of a network as suggested by the Examiner. The Examiner appears to have made such an analogy based on his personal knowledge.

Appellants have respectfully requested an affidavit or declaration from the Examiner indicating the basis for equating the claimed acyclic paths with the topology disclosed in Fitzgerald. To date, the Appellants have not received such a declaration or affidavit.

In sum, because Fitzgerald does not disclose each and every element of the claimed inventions, Fitzgerald cannot anticipate the claims of the present invention.

IX. CONCLUSION

Accordingly, for at least the aforementioned reasons, Appellants respectfully request the Honorable Members of the Board of Patent Appeals and Interferences to reverse each of the outstanding rejections in connection with

the present application and allow each of the pending claims 3-6 and 8-10 in connection with the present application.

If necessary, the Commissioner is hereby authorized in this, concurrent, and future replies, to charge payment or credit any overpayment to Deposit Account No.08-0750 for any additional fees required under 37 C.F.R. § 1.16 or under 37 C.F.R. § 1.17; particularly, extension of time fees.

Respectfully submitted,

HARNESS, DICKEY, & PIERCE, P.L.C.

By: _____

John E. Curtin, Reg. No. 37,602
P.O. Box 8910
Reston, Virginia 20195
(703) 668-8000

JEC:psy

APPENDIX A

1. (Canceled)

2. (Canceled)

3. (Previously Presented) A method of generating test sequences for evaluating the interoperability of communication systems connected through a first and second Voice-over-Internet Protocol (VoIP) gateway with respect to a desired mode of operation between the systems, the method comprising:

determining transitions that are required to implement the desired mode of operation, wherein each transition pertains to a first operation of the first VoIP gateway and a corresponding second operation of the second VoIP gateway;

said determining step further comprising,

generating acyclic paths from a transition diagram representing possible transitions;

generating simple cycles from said transition diagram; and

combining said generated paths and cycles to form a final set of paths representing said determined transitions; and

testing each communication system by causing each system to perform the determined transitions.

4. (Original) The method of claim 3, including providing at least one of the communication systems in the form of an Internet protocol network.

5. (Original) The method of claim 3, including providing at least one of the communication systems in the form of a switched telephone network.

6. (Original) The method of claim 3, including selecting the desired mode of communication as voice communication.

7. (Canceled)

8. (Previously Presented) The method of claim 3, including providing an Internet protocol (IP) network as one of the communication systems, coupling the first VoIP gateway system to the IP network, and coupling the second VoIP gateway system to the IP network.

9. (Previously Presented) The method of claim 8, including coupling a switched telephone network between the first and second VoIP gateway systems.

10. (Previously Presented) The method of claim 7, including eliminating from said testing step transitions concerning only the first VoIP gateway system, and transitions concerning only the second VoIP gateway system.

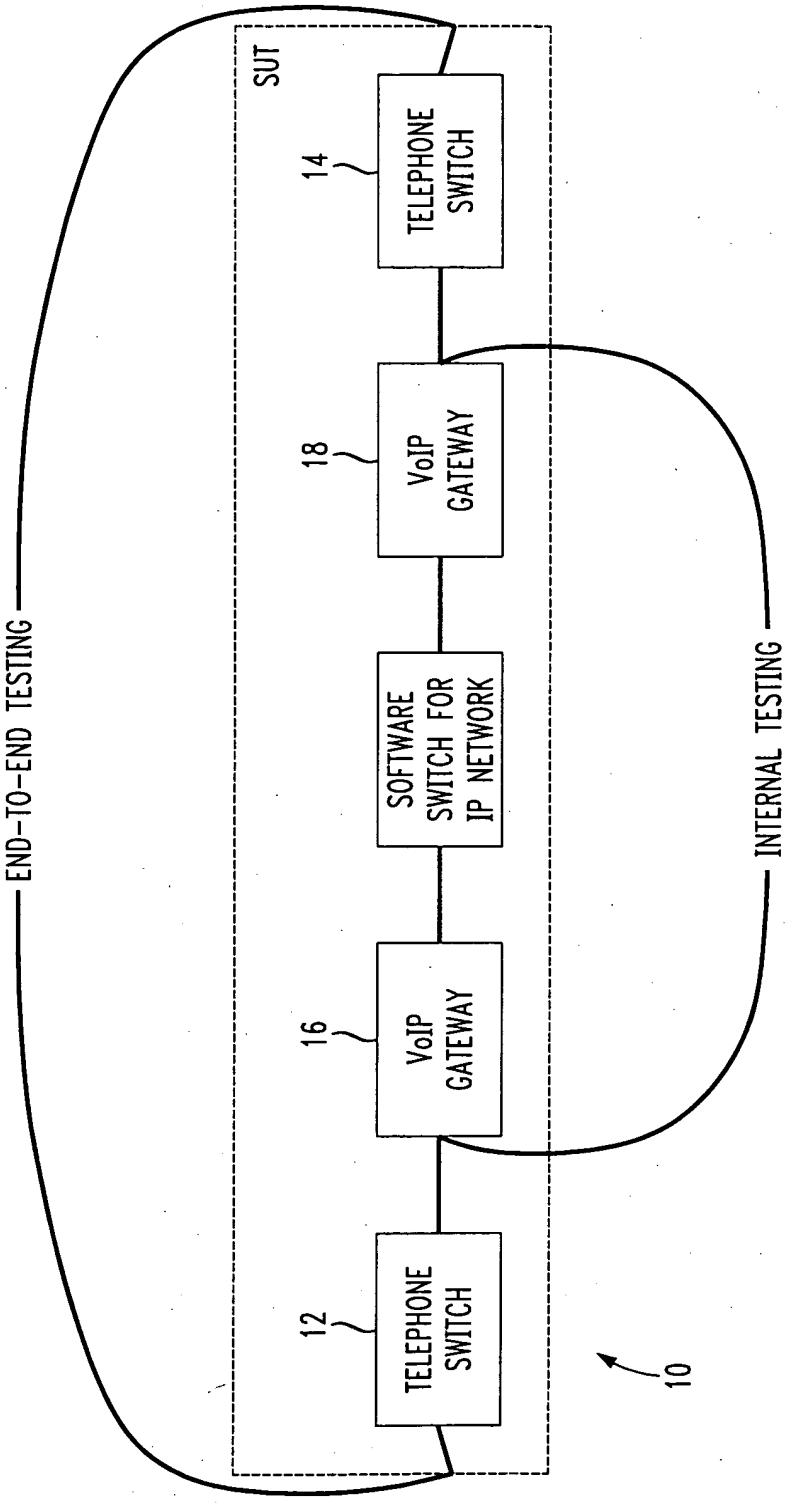
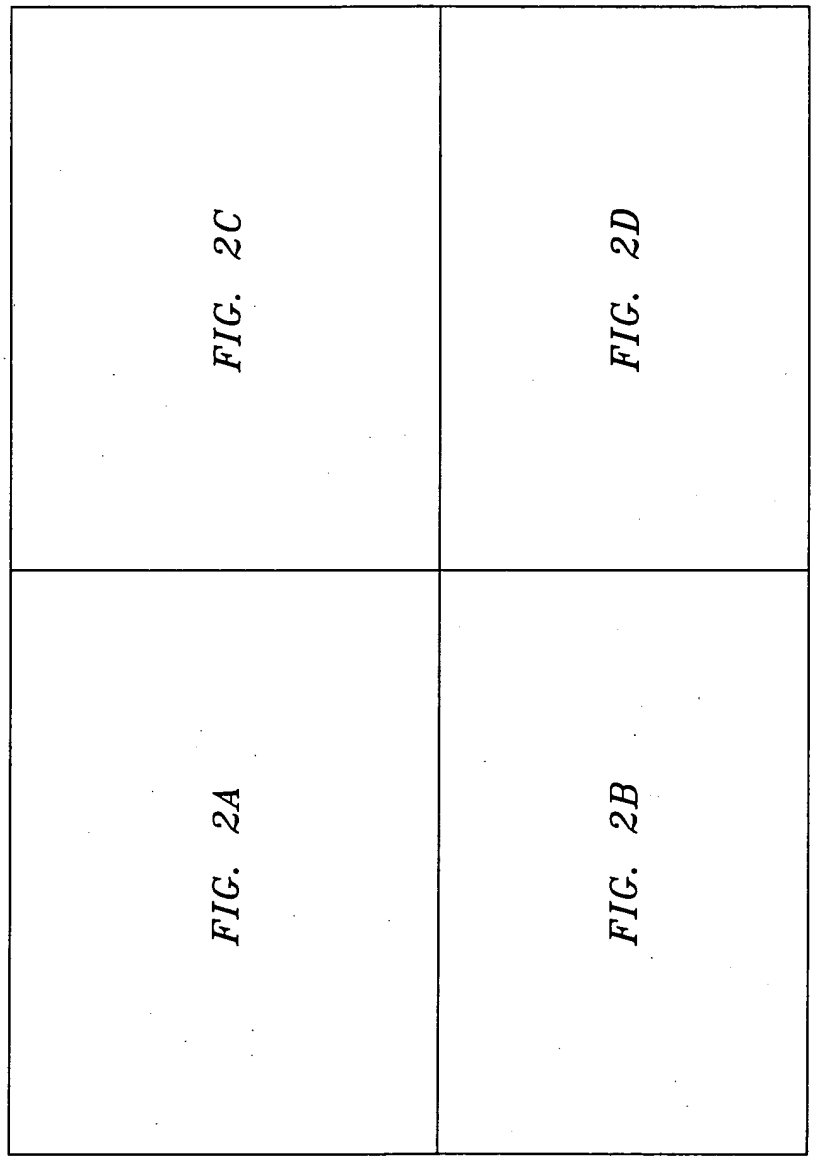

SUT
14
TELEPHONE SWITCH
18
VoIP GATEWAY
SOFTWARE SWITCH FOR IP NETWORK
16
VoIP GATEWAY
12
TELEPHONE SWITCH
10
END-TO-END TESTING
INTERNAL TESTING

FIG. 1

FIG. 2



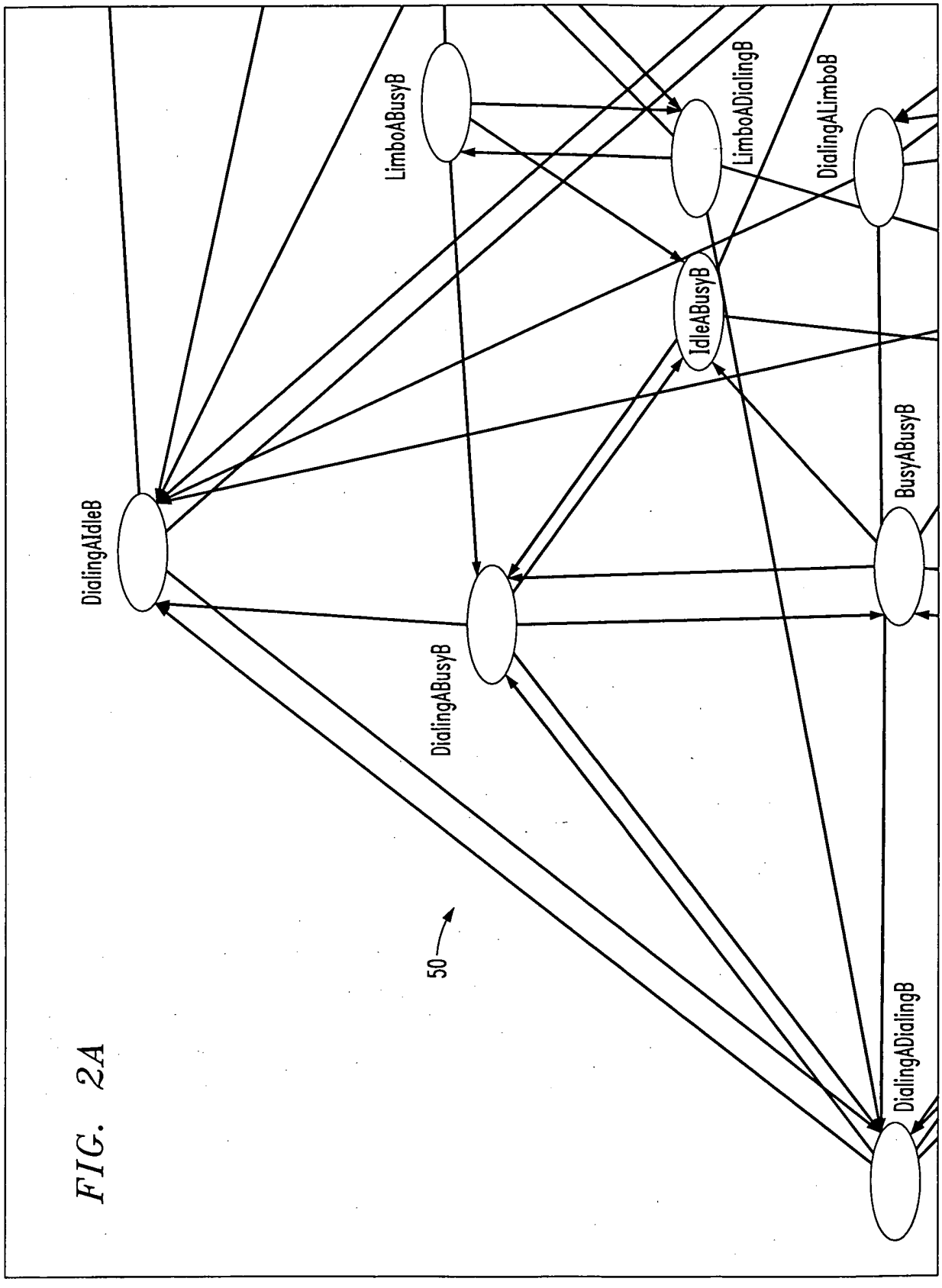

DialingAIdleB
LimboABusyB
LimboADialingB
DialingALimboB
IdleABusyB
BusyABusyB
DialingABusyB
DialingADialingB
50

FIG. 2A

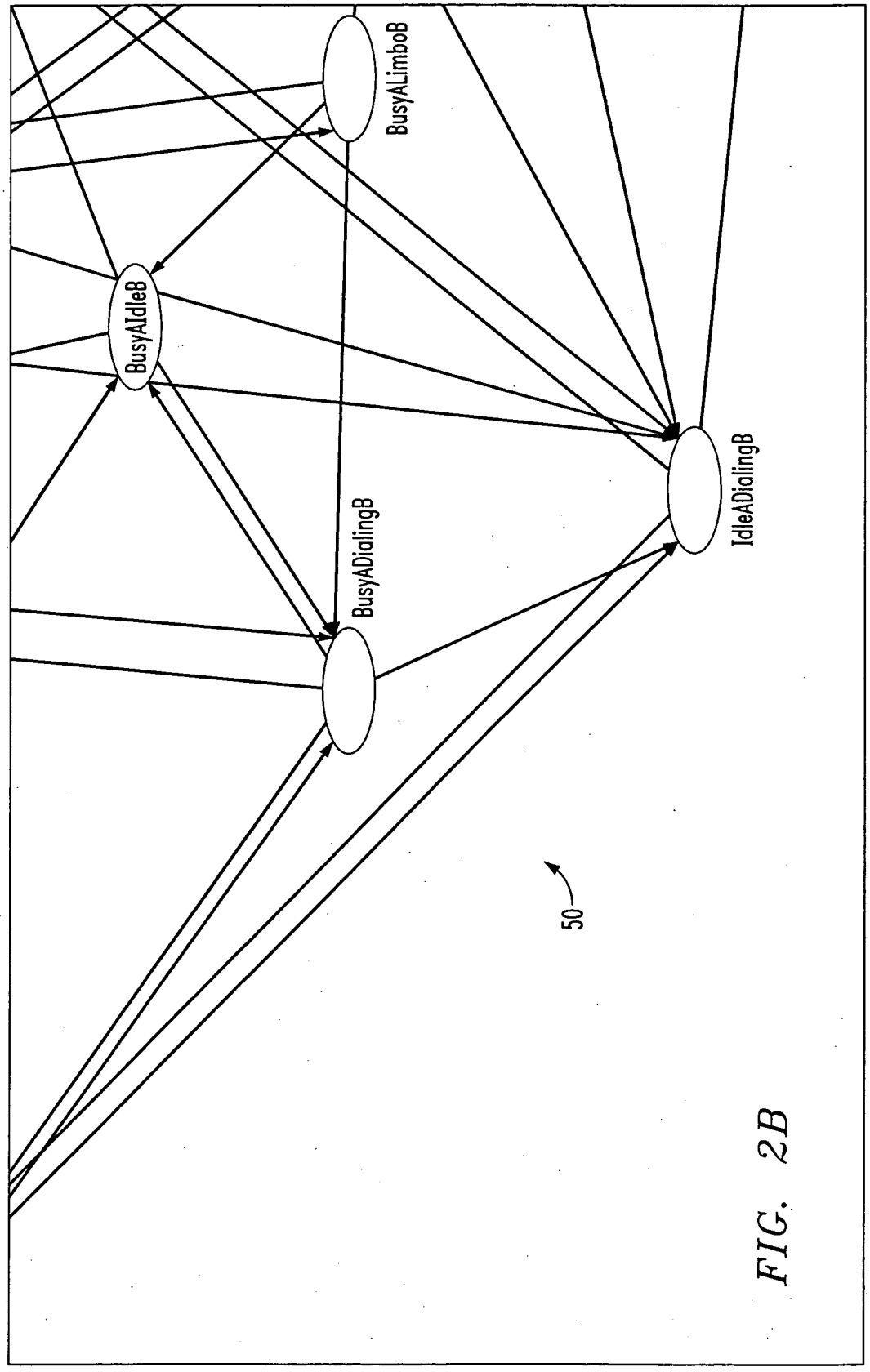
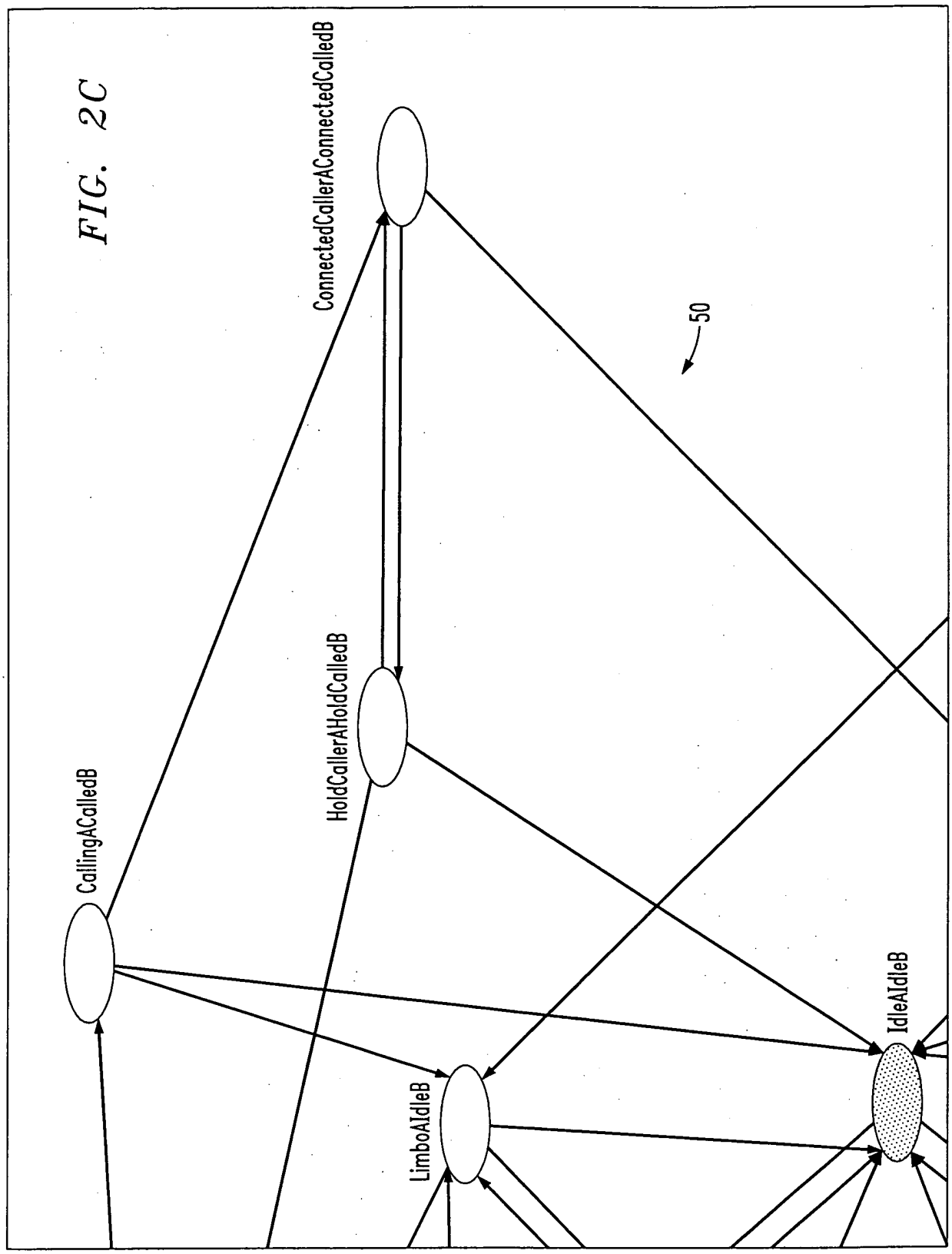


FIG. 2B

FIG. 2C


ConnectedCallerAConnectedCalledB
50
HoldCallerAHoldCalledB
CallingACalledB
IdleAIdleB
LimboAIdleB

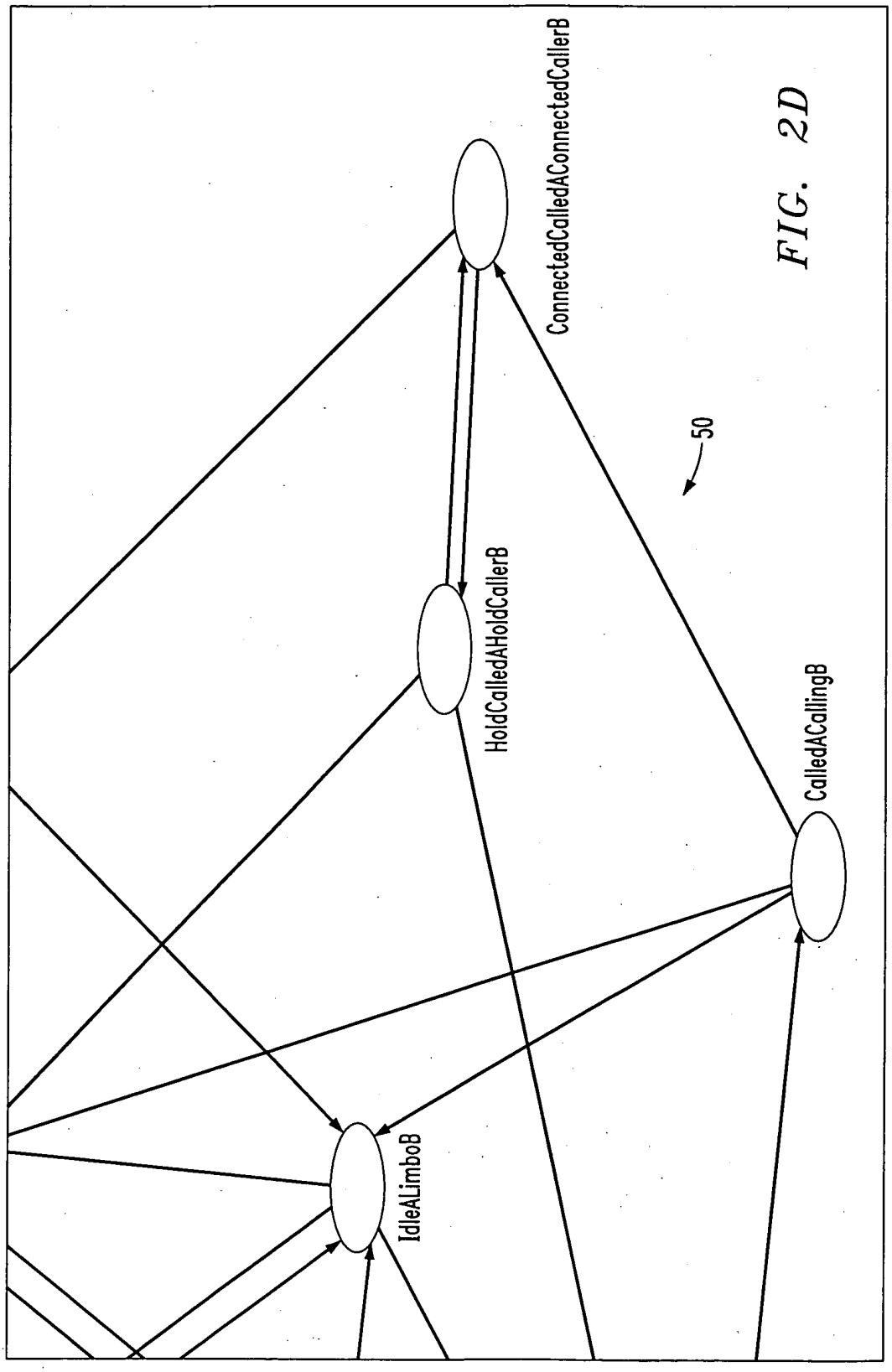

ConnectedCalledAConnectedCallerB
50
HoldCalledAHoldCallerB
CalledACallingB
IdleALimboB

FIG. 2D

FIG. 3

```
gui efsm
states { 21
 1.  CallingACalledB
 2.  DialingADialingB
 3.  IdleABusyB
 4.  CalledACallingB
 5.  BusyABusyB
 6.  DialingALimboB
 7.  BusyADialingB
 8.  BusyAIdleB
 9.  BusyALimboB
10.  LimboADialingB
11.  ConnectedCalledAConnectedCallerB
12.  LimboABusyB
13.  IdleADialingB
14.  LimboAIdleB
15.  DialingABusyB
16.  HoldCalledAHoldCallerB
17.  HoldCallerAHoldCalledB
18.  DialingAIdleB
19.  IdleALimboB
20.  ConnectedCallerAConnectedCalledB
21.  IdleAIdleB
}
```

i = input
o = output
p = predicate
a = action

```
start {
IdleAIdleB

boolean term=false;
boolean IdleA=true;
boolean IdleB=true;
boolean DialingA=false;
boolean DialingB=false;

}

transitions { 68
1.   BusyADialingB BusyAIdleB {
        i { {On-hook B} }
        o { {} }
        p { true }
        a { { DialingB=false; IdleB=true } }
     }
2.   IdleADialingB DialingADialingB {
        i { {Off-hook A} }
        o { {DialTone A} }
        p { true }
        a { {IdleA=false; DialingA=true} }
     }
3.   BusyADialingB IdleADialingB {
        i { {On-hook A} }
        o { {} }
        p { true }
        a { { IdleA=true } }
     }
4.   IdleABusyB IdleAIdleB {
        i { {On-hook B} }
        o { {} }
        p { true }
        a { {term=true;IdleB=true} }
```

FIG. 4

```
}
5. HoldCallerAHoldCalledB IdleAIdleB {
     i { {On-hook A} }
     o { {} }
     p { true }
     a { {term=true;IdleA=true;IdleB=true} }
}
6. IdleALimboB IdleADialingB {
     i { {Timeout B} }
     o { {DialTone B} }
     p { true }
     a { { DialingB=true } }
}
7. BusyADialingB BusyABusyB {
     i { {Dial B A} }
     o { {LineBusyTone B} }
     p { true }
     a { { DialingB=false } }
}
8. HoldCalledAHoldCallerB IdleAIdleB {
     i { {On-hook B} }
     o { {} }
     p { true }
     a { {term=true;IdleB=true;IdleA=true} }
}
9. DialingAIdleB DialingADialingB {
     i { {Off-hook B} }
     o { {DialTone B} }
     p { true }
     a { { IdleB=false; DialingB=true } }
}
10. IdleADialingB IdleAIdleB {
     i { {On-hook B} }
     o { {} }
     p { true }
     a { {term=true;IdleB=true;DialingB=false} }
}
11. BusyABusyB BusyAIdleB {
     i { {On-hook B} }
     o { {} }
     p { true }
     a { { IdleB=true } }
}
12. DialingADialingB BusyADialingB {
     i { {Dial A B} }
     o { {LineBusyTone A} }
     p { true }
     a { DialingA=false }
}
13. IdleAIdleB IdleADialingB {
     i { {Off-hook B} }
     o { {DialTone B} }
     p { !term }
     a { {IdleB=false; DialingB=true} }
}
14. HoldCalledAHoldCallerB IdleADialingB {
     i { {Timeout AB} }
     o { {DialToneB} }
     p { true }
     a { { IdleA=true; DialingB=true } }
}
15. IdleABusyB DialingABusyB {
     i { {Off-hook A} }
     o { {DialTone A} }
     p { true }
     a { { IdleA=false; DialingA=true } }
}
16. CallingACalledB ConnectedCallerAConnectedCalledB {
     i { {Off-hook B} }
     o { {} }
     p { true }
     a { {} }
```

FIG. 5

```
    }
17. DialingADialingB DialingABusyB {
        i { {Dial B A} }
        o { {LineBusyTone B} }
        p { true }
        a { { DialingB=false } }
    }
18. CalledACallingB IdleALimboB {
        i { {Timeout AB} }
        o { {} }
        p { true }
        a { { IdleA=true } }
    }
19. ConnectedCallerAConnectedCalledB IdleALimboB {
        i { {On-hook A} }
        o { {} }
        p { true }
        a { { IdleA=true } }
    }
20. ConnectedCallerAConnectedCalledB HoldCallerAHoldCalledB {
        i { {On-hook B} }
        o { {} }
        p { true }
        a { {} }
    }
21. IdleAIdleB DialingAIdleB {
        i { {Off-hook A} }
        o { {DialTone A} }
        p { !term }
        a { {IdleA=false; DialingA=true} }
    }
22. DialingABusyB BusyABusyB {
        i { {Dial A B} }
        o { {LineBusyTone A} }
        p { true }
        a { { DialingA=false } }
    }
23. IdleALimboB IdleAIdleB {
        i { {On-hook B} }
        o { {} }
        p { true }
        a { {term=true;IdleB=true} }
    }
24. LimboAIdleB IdleAIdleB {
        i { {On-hook A} }
        o { {} }
        p { true }
        a { {term=true;IdleA=true} }
    }
25. HoldCalledAHoldCallerB ConnectedCalledAConnectedCallerB {
        i { {Off-hook A} }
        o { {} }
        p { true }
        a { {} }
```

FIG. 6

```
    }
26. DialingABusyB IdleABusyB {
        i { {On-hook A} }
        o { {} }
        p { true }
        a { { DialingA=false; IdleA=true } }
    }
27. DialingABusyB DialingAIdleB {
        i { {On-hook B} }
        o { {} }
        p { true }
        a { { IdleB=true } }
    }
28. HoldCallerAHoldCalledB DialingAIdleB {
        i { {Timeout AB} }
        o { {DialTone A} }
        p { true }
        a { { DialingA=true; IdleB=true } }
    }
29. DialingAIdleB CallingACalledB {
        i { {Dial A B} }
        o { {AudibleRinging A,Ringing B} }
        p { true }
        a { {DialingA=false; IdleB=false} }
    }
30. IdleADialingB CalledACallingB {
        i { {Dial B A} }
        o { {AudibleRinging B,Ringing A} }
        p { true }
        a { { IdleA=false; DialingB=false } }
    }
31. BusyABusyB IdleABusyB {
        i { {On-hook A} }
        o { {} }
        p { true }
        a { { IdleA=true } }
    }
32. CalledACallingB IdleAIdleB {
        i { {On-hook B} }
        o { {} }
        p { true }
        a { {term=true;IdleA=true;IdleB=true} }
    }
33. ConnectedCalledAConnectedCallerB HoldCalledAHoldCallerB {
        i { {On-hook A} }
        o { {} }
        p { true }
        a { {} }
    }
34. BusyAIdleB BusyADialingB {
        i { {Off-hook B} }
        o { {DialTone B} }
        p { true }
        a { { IdleB=false; DialingB=true } }
    }
35. LimboAIdleB DialingAIdleB {
        i { {Timeout A} }
        o { {DialToneA} }
        p { true }
        a { { DialingA = true; } }
    }
36. IdleALimboB DialingALimboB {
        i { {Off-hook A} }
        o { {DialTone A} }
        p { true }
        a { {IdleA=false;DialingA=true} }
    }
37. LimboAIdleB LimboADialingB {
        i { {Off-hook B} }
        o { {DialTone B} }
        p { true }
        a { {IdleB=false;DialingB=true} }
```

FIG. 7

```
}
38. DialingALimboB IdleALimboB {
      i { {On-hook A} }
      o { {} }
      p { true }
      a { {IdleA=true;DialingA=false} }
}
39. BusyALimboB IdleALimboB {
      i { {On-hook A} }
      o { {} }
      p { true }
      a { IdleA=true }
}
40. LimboABusyB IdleABusyB {
      i { {On-hook A} }
      o { {} }
      p { true }
      a { IdleA=true }
}
41. LimboADialingB IdleADialingB {
      i { {On-hook A} }
      o { {} }
      p { true }
      a { IdleA=true }
}
42. DialingALimboB DialingAIdleB {
      i { {On-hook B} }
      o { {} }
      p { true }
      a { IdleB=true }
}
43. DialingAIdleB IdleAIdleB {
      i { {On-hook A} }
      o { {} }
      p { true }
      a { {term=true;IdleA=true;DialingA=false} }
}
44. BusyALimboB BusyAIdleB {
      i { {On-hook B} }
      o { {} }
      p { true }
      a { IdleB=true }
}
45. BusyAIdleB IdleAIdleB {
      i { {On-hook A} }
      o { {} }
      p { true }
      a { {term=true;IdleA=true} }
}
46. LimboADialingB LimboAIdleB {
      i { {On-hook B} }
      o { {} }
      p { true }
      a { {DialingB=false;IdleB=true} }
}
47. LimboABusyB LimboAIdleB {
      i { {On-hook B} }
      o { {} }
      p { true }
      a { IdleB=true }
}
48. CalledACallingB ConnectedCalledAConnectedCallerB {
      i { {Off-hook A} }
      o { {} }
      p { true }
      a { {} }
}
49. DialingALimboB BusyALimboB {
      i { {Dial A B} }
      o { {LineBusyTone A} }
      p { true }
      a { DialingA=false }
```

FIG. 8

```
}
50. DialingADialingB IdleADialingB {
        i { {On-hook A} }
        o { {} }
        p { true }
        a { {IdleA=true; DialingA=false} }
    }
51. LimboADialingB LimboABusyB {
        i { {Dial B A} }
        o { {LineBusyTone B} }
        p { true }
        a { DialingB=false }
    }
52. CallingACalledB LimboAIdleB {
        i { {Timeout AB} }
        o { {} }
        p { true }
        a { { IdleB=true; } }
    }
53. BusyAIdleB DialingAIdleB {
        i { {Timeout A} }
        o { {DialTone A} }
        p { true }
        a { DialingA=true }
    }
54. DialingADialingB DialingAIdleB {
        i { {On-hook B} }
        o { {} }
        p { true }
        a { { DialingB=false; IdleB=true } }
    }
55. BusyADialingB DialingADialingB {
        i { {Timeout A} }
        o { {DialTone A} }
        p { true }
        a { DialingA=true }
    }
56. ConnectedCalledAConnectedCallerB LimboAIdleB {
        i { {On-hook B} }
        o { {} }
        p { true }
        a { { IdleB=true } }
    }
57. BusyALimboB DialingALimboB {
        i { {Timeout A} }
        o { {DialTone A} }
        p { true }
        a { DialingA=true }
    }
58. HoldCallerAHoldCalledB ConnectedCallerAConnectedCalledB {
        i { {Off-hook B} }
        o { {} }
        p { true }
        a { {} }
    }
59. BusyABusyB DialingABusyB {
        i { {Timeout A} }
        o { {DialTone A} }
        p { true }
        a { DialingA=true }
    }
60. CallingACalledB IdleAIdleB {
        i { {Onhook A} }
        o { {} }
        p { true }
        a { {term=true;IdleA=true;IdleB=true} }
    }
61. LimboADialingB DialingADialingB {
        i { {Timeout A} }
        o { {DialTone A} }
        p { true }
        a { DialingA=true }
```

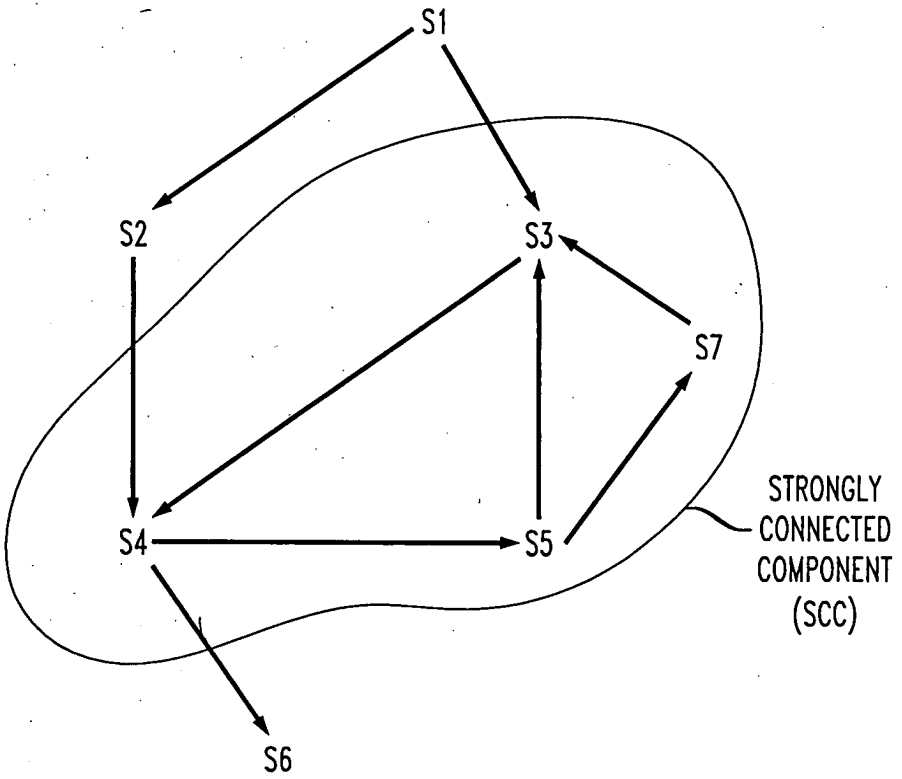
FIG. 9

```
    }
62. LimboABusyB DialingABusyB {
        i { {Timeout A} }
        o { {DialTone A} }
        p { true }
        a { DialingA=true }
    }
63. IdleABusyB IdleADialingB {
        i { {Timeout B} }
        o { {DialTone B} }
        p { true }
        a { DialingB=true }
    }
64. DialingABusyB DialingADialingB {
        i { {Timeout B} }
        o { {DialTone B} }
        p { true }
        a { DialingB=true }
    }
65. DialingALimboB DialingADialingB {
        i { {Timeout B} }
        o { {DialTone B} }
        p { true }
        a { DialingB=true }
    }
66. BusyABusyB BusyADialingB {
        i { {Timeout B} }
        o { {DialTone B} }
        p { true }
        a { DialingB=true }
    }
67. BusyALimboB BusyADialingB {
        i { {Timeout B} }
        o { {DialTone B} }
        p { true }
        a { DialingB=true }
    }
68. LimboABusyB LimboADialingB {
        i { {Timeout B} }
        o { {DialTone B} }
        p { true }
        a { DialingB=true }
    }
}
```

FIG. 10

1 GENERATE ALL ACYCLIC PATHS

1a. COLLAPSE STRONGLY CONNECTED COMPONENTS INTO A SINGLE NODE



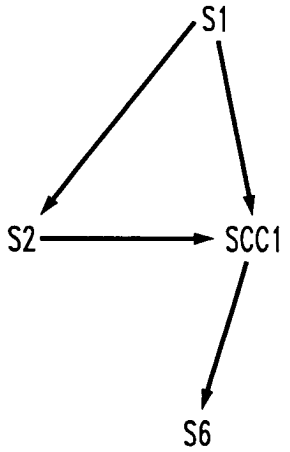
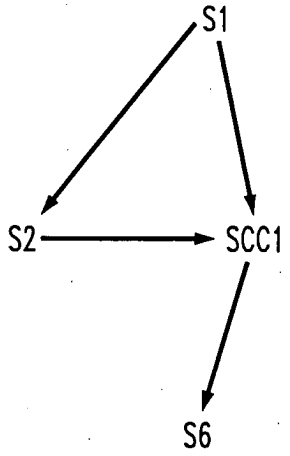


FIG. 11

1 GENERATE ALL ACYCLIC PATHS

1b DETERMINE ALL PATHS IN DAG:



S1, S2, SCC1, S6
S1, SCC1, S6

FIG. 12

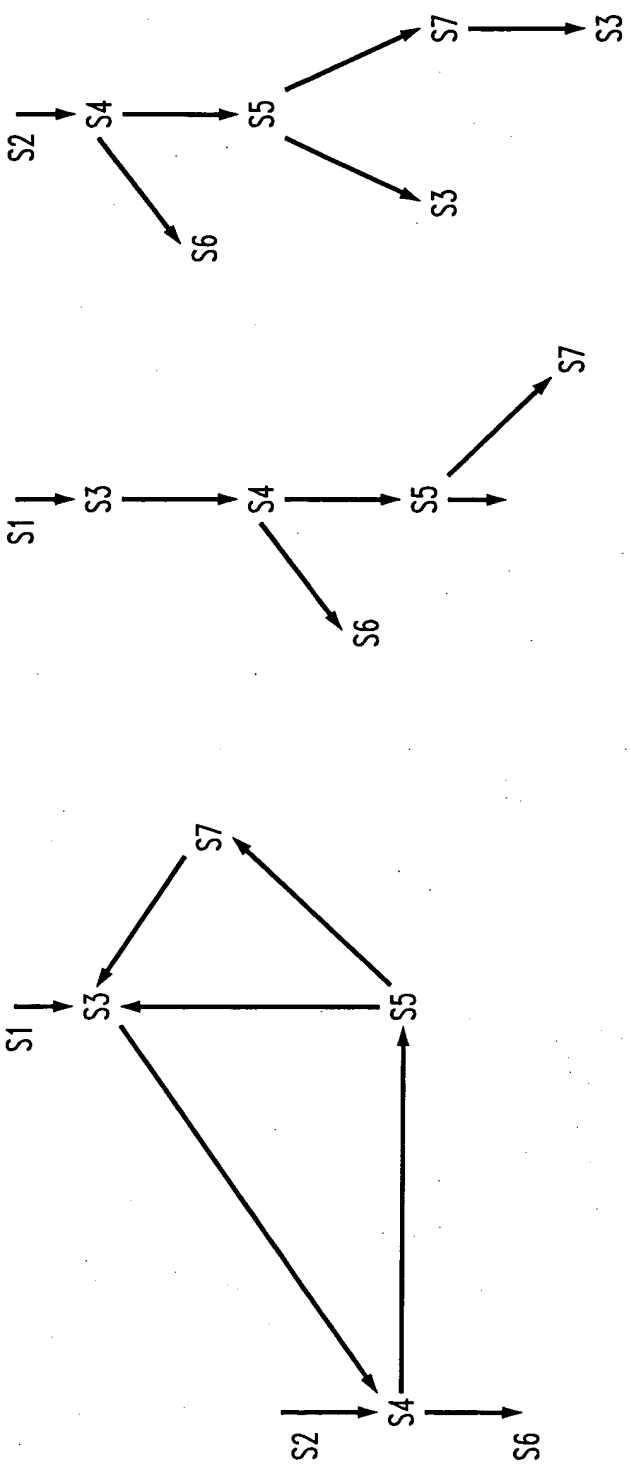
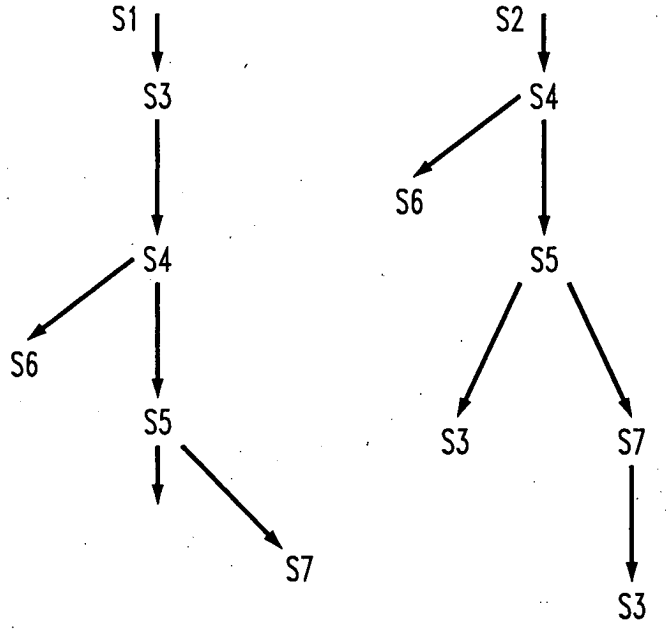

1 GENERATE ALL ACYCLIC PATHS
1c BUILD NEXT-TRANSITION TREE FOR SCC:
S1
S3
S7
S5
S2
S4
S6
S1
S3
S4
S5
S7
S6
S2
S4
S5
S7
S3
S6
S3

FIG. 13

1 GENERATE ALL ACYCLIC PATHS

1d EXPAND EACH SCC WITH ALL PATHS THROUGH NEXT-TRANSITION TREE



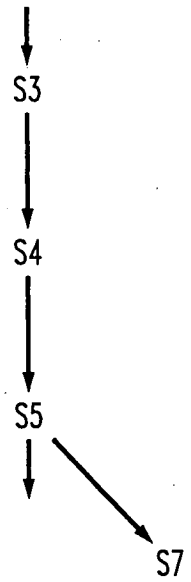
S1, S2, SCC1, S6
-> S1, S2, S4, S6

S1, SCC1, S6
-> S1, S3, S4, S6

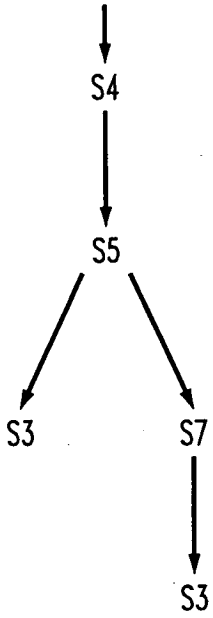
FIG. 14

2 GENERATE ALL SIMPLE CYCLES

RE-USE NEXT-TRANSITION TREE



S3, S4, S5, S3
S3, S4, S5, S7, S3



S4, S5, S3, S4
S4, S5, S7, S3, S4

FIG. 15

3 COMBINE THE PATHS AND THE CYCLES

S1, S2, S4, S6 S1, S3, S4, S6	+	S3, S4, S5, S3 S3, S4, S5, S7, S3

=

S1, S2, S4, S5, S3, S4, S6
S1, S2, S4, S5, S7, S3, S4, S6
S1, S3, S4, S5, S3, S4, S6
S1, S3, S4, S5, S7, S3, S4, S6

FIG. 16

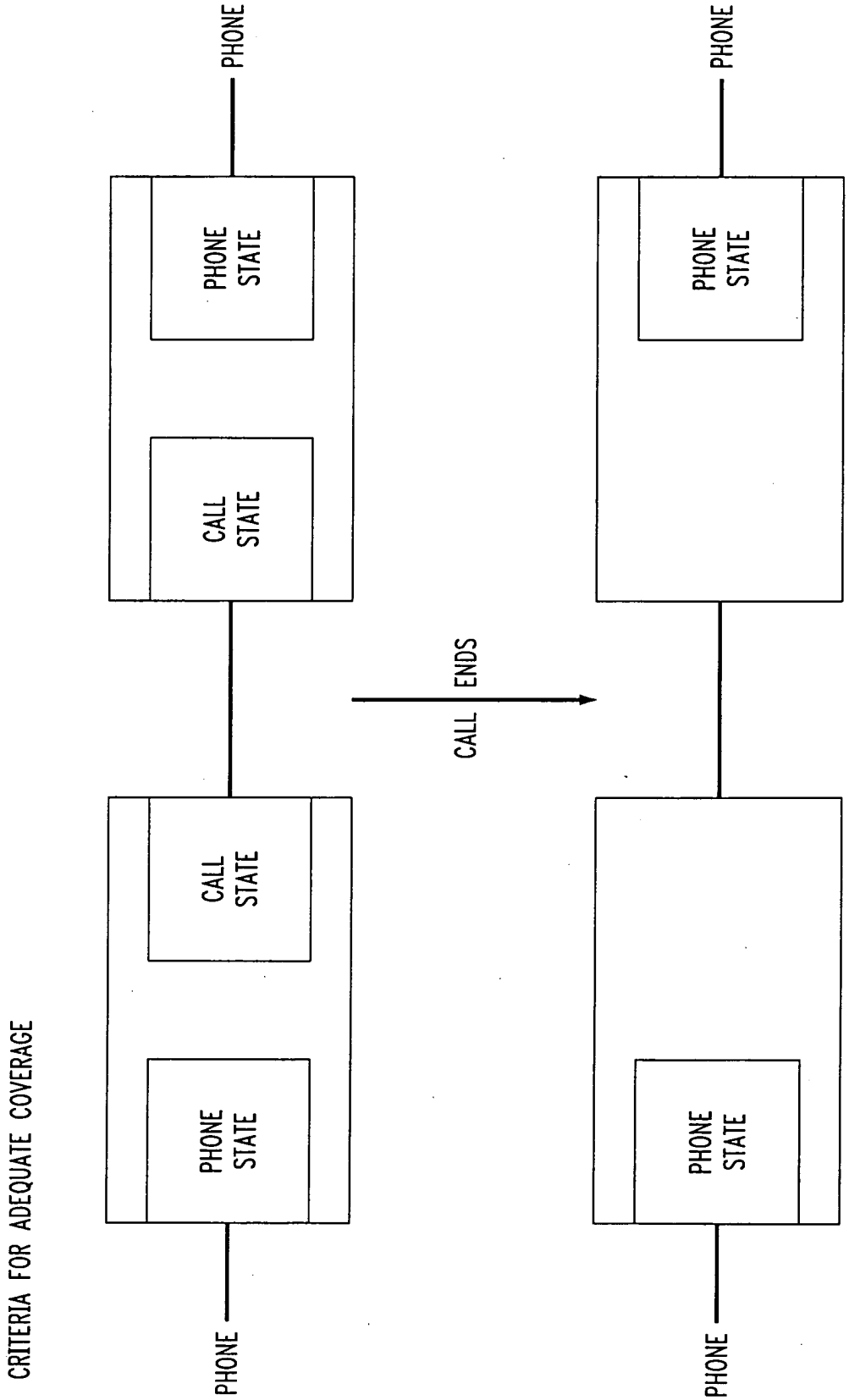
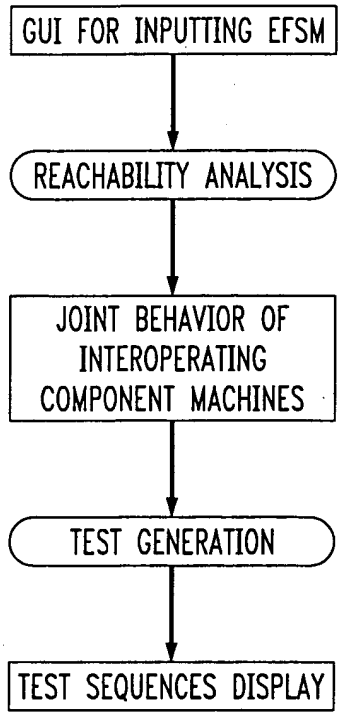

CRITERIA FOR ADEQUATE COVERAGE
PHONE
PHONE STATE
CALL STATE
CALL STATE
PHONE STATE
PHONE
CALL ENDS
PHONE
PHONE STATE
PHONE STATE
PHONE

FIG. 17


GUI FOR INPUTTING EFSM
REACHABILITY ANALYSIS
JOINT BEHAVIOR OF INTEROPERATING COMPONENT MACHINES
TEST GENERATION
TEST SEQUENCES DISPLAY